Simplifying very deep convolutional neural network architectures for robust speech recognition

SIMPLIFYING VERY DEEP CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES FOR ROBUST SPEECH RECOGNITION

Joanna Rownicka, Steve Renals, Peter Bell

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom

ABSTRACT

Very deep convolutional neural networks (VDCNNs) have been successfully used in computer vision. More recently VDCNNs have been applied to speech recognition, using architectures adopted from computer vision. In this paper, we experimentally analyse the role of the components in VDCNN architectures for robust speech recognition. We have proposed a number of simplified VDCNN architectures, taking into account the use of fully-connected layers and down-sampling approaches. We have investigated three ways to down-sample feature maps: max-pooling, average-pooling, and convolution with increased stride. Our proposed model consisting solely of convolutional (conv) layers, and without any fully-connected layers, achieves a lower word error rate on Aurora 4 compared to other VDCNN architectures typically used in speech recognition. We have also extended our experiments to the MGB-3 task of multi-genre broadcast recognition using BBC TV recordings. The MGB-3 results indicate that the same architecture achieves the best result among our VDCNNs on this task as well.

Index Terms— Robust Speech Recognition, Very Deep Convolutional Neural Networks, Aurora 4, MGB Challenge

1. INTRODUCTION

Convolutional neural networks (CNNs) were the one of the first successfully used deep neural network architectures, originally used for image processing, computer vision, and document understanding [1, 2], and since about 2012 they have defined the state-of-the-art for many computer vision tasks [3, 4]. CNNs take account of the local input structure by using local receptive fields to model spatially local correlations. Combining this local modeling with weight sharing and pooling enables invariances to be exploited across the structure of the input, typically leading to better generalization to unseen data compared to fully-connected deep neural networks (DNNs).

CNNs have been applied to speech recognition by treating time-frequency representations analogously to images [5, 6, 7, 8, 9, 10], and also through learning one-dimension structure of sequences, an approach referred to the time-delay neural network (TDNN) [11, 12]. There is experimental evidence that convolutional structure of CNNs allows better solutions to be learned compared with DNNs. For instance, Huang et al. [9] estimated both CNN and DNN acoustic models on a speech recognition task, training both networks on about 1 000 h of data – an amount possibly big enough for a DNN to learn all the necessary invariances. The CNN resulted in reduced word error rates compared to the DNN; moreover for a distant speech recognition task the gain of CNNs over DNNs increased in direct proportion to the speaker-microphone distance.

Very deep convolutional neural networks (VDCNNs) have been shown to improve the recognition accuracy compared to CNNs with fewer layers for both image recognition [13, 14] and speech recognition [15, 16, 17, 18]. The key concept of VDCNNs is to replace a single kernel from a classical CNN model with a stack of convolutional kernels of smaller size. Such a stack of small kernels can have a similar effective receptive field as a single larger kernel but with a reduced number of independent parameters. Moreover, stacking convolutional layers enables more complex features to be learned due to the additional non-linearities in the model.

Simplifying a typical CNN architecture (with pooling and fully-connected layers) to an architecture consisting solely of convolutional layers has been shown to achieve state-of-the-art accuracy on several image recognition tasks – for instance, GoogLeNet [14], is a convolutional network which uses average pooling instead of fully-connected layers, eliminating the majority of trainable parameters of the network without reducing object recognition accuracy. Deep residual learning (ResNet) [19], is another example of a VDCNN without fully-connected layers at the top of the network.

The use of VDCNNs for noise-robust speech recognition was investigated by Qian and Woodland [18] for the Aurora 4 robust speech recognition and AMI distant speech recognition corpora, with a specific focus on optimal VDCNN architectures – the kernel sizes, pooling and padding strategies, and the size of the input feature map. Most VDCNN acoustic models have used architectures in which the upper hidden layers are fully-connected [15, 16, 18]; in contrast, Yu et al. [17] use an architecture without any fully-connected layers. Furthermore, they also do not employ pooling layers – convolutional layers with a larger stride are used to downsample the feature maps. However Yu et al. compare the resultant acous-

tic model architecture to DNN and long short-term memory (LSTM) recurrent neural network (RNN) models, with no comparison across CNN architectures, which makes it difficult to assess the contribution of the novel CNN architectural components that they use – it is not clear whether removing fully-connected layers and replacing pooling layers with convolutional layers improves or degrades the accuracy of the speech recognition system.

In this paper we investigate VDCNNs to find out which components are necessary to achieve the state-of the art accuracy for robust speech recognition. Specifically, we want to learn the role and the importance of different components of the network (fully-connected layers, pooling layers, convolution stride) in noisy speech recognition applied to the Aurora 4 and multi-genre broadcast (MGB) data sets.

2. CONVOLUTIONAL NEURAL NETWORKS FOR SPEECH RECOGNITION

In comparison to other state-of-the-art neural network acoustic models, such as RNNs/LSTMs, CNNs are often easier to train because of their feed-forward nature. Moreover, the key properties of CNN models (locality, weight sharing, and pooling) can contribute to improved noise robustness and reduced overfitting of the models.

2.1. Acoustic features, locality and weight sharing

The feature map input to CNN acoustic models typically employs a sub-sequence of log mel filterbank (FBANK) features (often concatenated with their first and second temporal derivatives) arranged in a two-dimension array (feature map) whose size is $time \times freq$ (where $time$ is the width of the time context window and $freq$ is the number of frequency bins in the FBANK). FBANK features are more suitable for CNN models than decorrelated features such as mel-frequency cepstral coefficients (MFCCs).

CNNs with FBANK inputs can exploit the local spectral representation structure when learning the weights of the hidden layers. For robust speech recognition, this locality can enable the network to model the situation in which noise or distortion is more apparent in some bands of the spectrum than in others, allowing representations to be computed from the cleaner parts of the spectrum. The shared weight structure of a convolutional layer enables translation invariance to be captured, by extracting the same feature at all points of the input feature map. Although this is less important for acoustic modeling, compared with image recognition, it does enable some robustness to different speakers and speaking styles which may be manifested as variations in activity in different frequency bands. Moreover, weight sharing and the use of local receptive fields reduces the number of the parameters of the model which can contribute to the reduction in overfitting of the model.

2.2. Convolutional layer

Convolutional layers are based on a weighted sum of their inputs, similar to fully-connected (FC) layers, with the difference that hidden units in the convolutional layer are connected only to a subregion of the input feature map (the receptive field) and the hidden units share weights across the local receptive fields in order to extract the same feature across the input representation. Each feature map hidden unit value $h_{i,j}$ in a CNN model is computed as

$$h_{i,j} = \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} f(w_{k,l} x_{i+k,j+l} + b_{i,j}) \quad (1)$$

$$= f(\boldsymbol{W} \otimes \boldsymbol{X} + b_{i,j}), \quad (2)$$

where f is the activation function, $w_{k,l}$ are items of the shared $m \times m$ weight matrix $\boldsymbol{W}$, $b_{i,j}$ is the shared bias, and $x_{i+k,j+l}$ is the input at position $i+k, j+l$. We use k and l to index into the receptive field, whose top left corner is at $x_{i,j}$. $\boldsymbol{X}$ denotes a matrix of input values within local receptive field. $\otimes$ denotes cross-correlation and this is the operation performed by convolutional layers in our models.[1]

2.3. Padding

Padding input representations with zeros around the border enables the size of the output feature map to match the size of the input feature map, which is important for deep convolutional architectures; moreover it ensures that information is not lost from the edges of the input feature map. This is sometimes referred to as SAME padding, in contrast to a valid convolution which does not use padding and thus results in a smaller output feature map.

Shallower CNN acoustic models have tended to use valid convolution [5, 6, 8], however more recently investigated VDCNN models [16, 17, 18] have used SAME padding to preserve the feature map size.

2.4. Downsampling

CNN classifiers require some form of compression or downsampling to map a feature map to a classification. This is most often achieved through the use of pooling layers between convolutional layers. Pooling layers discard the exact positional information of a feature, and may be regarded as smoothing filters. In practice two types of pooling operations are used in CNN models: max and average pooling. Max pooling is sensitive to the existence of some feature in subregion of the initial representation, and average pooling measures the

[1] If the cross-correlation kernel $\boldsymbol{w}$ is flipped horizontally and vertically, then (2) becomes a convolution. The network learns the kernel appropriate to its orientation – so if convolution is implemented with a flipped kernel, it will learn that it is a flipped representation. The specific properties of convolution but not of cross-correlation (commutativity and associativity) are not required for a CNN acoustic model.

mean value of existence of a feature in that subregion. In practice, max pooling has shown better empirical results than average pooling in many pattern recognition tasks. Alternatively, a feature map can be downsampled using a convolutional layer with a larger stride. For image recognition, it has been demonstrated that replacing a max-pooling layer by a convolutional layer with increased stride does not reduce the classification accuracy [20]. Using convolutional layers instead of pooling layers to downsample feature maps can also be seen as learning the pooling operation rather than fixing it.

Both these approaches to downsampling maintain translation invariance and offer a degree of smoothing, and in a speech recognition context can offer robustness by compensating for variation in the frequency domain arising from the acoustic environment or speaker characteristics. In addition downsampling can also reduce the complexity of the model (although downsampling is often accompanied by an increase in the number of filters). Pooling layers have been used more frequently than convolution layers for downsampling in CNN acoustic modeling; however Yu et al. [17] downsampled using convolutional layers with stride $s = 2$ in both time and frequency dimensions in their VDCNN acoustic model.

3. EXPERIMENTAL SETUP

We use a hybrid neural network (NN) – hidden Markov model (HMM) speech recognition system. A Gaussian Mixture Model (GMM) based HMM system is first trained to estimate the context-dependent phone models, from which a forced alignments is obtained which provides the targets for training all our NN-HMM systems. We use Kaldi [21] to train the GMM-HMM system with $\sim$3.4k (for Aurora 4) or $\sim$9.4k (for MGB) clustered states using maximum likelihood estimation criterion and the standard Kaldi MFCC-LDA-MLLT-FMLLR features. In Aurora 4, decoding was performed with the task-standard bigram language model. For MGB, a trigram language model was used, with lattices being later rescored with a 4-gram language model [22].

In all of our experiments with neural network acoustic models we used FBANK features as inputs. Following the fundings in [18], where using only static features was the best setup for robust speech recognition, we also use only one feature map at the input. We use 40 filters and 5 frames of context, so the input feature map size is 11×40.

We used the TensorFlow software framework [23] to train the NN models and the tfkaldi software [24] to integrate TensorFlow neural networks with Kaldi.

3.1. Datasets

To train and evaluate our simplified VDCNN architectures for the task of robust speech recognition we use the Aurora 4 [25] and MGB Challenge [22] datasets.

3.1.1. Aurora 4

Aurora 4 is a medium-vocabulary task based on the Wall Street Journal (WSJ) corpus. The training set contains 7 138 utterances from 83 speakers. The clean-condition training set is equivalent to the SI-84 WSJ corpus with an overall audio duration of about 15 h. The multi-condition training set was designed to study the effects of variation in microphone and noise. Its overall duration is equal to the clean-condition training set duration. It consists of selected utterances from the clean-condition training set (recorded with the primary Sennheiser microphone) and utterances from 18 different microphone types (with the same linguistic content). 75% of utterances recorded with a primary microphone were corrupted using six noise types at different SNR levels (10-20 dB). The same noise distortion is applied to the mismatched microphone subset. Unless otherwise stated, in our experiments we use the GMM-HMM forced alignments generated from multi-condition training set.

The test set (test_eval92) was grouped into 4 subsets: 330 clean utterances selected from SI-84 WSJ corpus, 330×6 utterances with 6 types of additive noise at 5-15 dB SNR, 330 utterances recorded with a different microphone, 330×6 utterances recorded with a different microphone and with 6 types of additive noise at 5-15 dB SNR. Those test set subsets are referred to as A, B, C, and D, respectively. The overall test set audio duration for all four subsets was about 9 h.

3.1.2. MGB-3 and MGB-1

The MGB datasets consist of multi-genre BBC English TV recordings from the MGB Challenge competitions in 2015 (MGB-1) and 2017 (MGB-3). In this paper we train models on the MGB-3 training set, which contains recordings from around 750 episodes (about 350 hours). Note that it is substantially smaller than the original MGB-1 training set. We evaluate our models on both the MGB-3 development set and MGB-1 test set. The MGB-3 dev set consists of about 5 h and MGB-1 test set about 19 h of multi-genre TV episodes.

3.2. Architectures and training

Fig. 1 shows the architectures of the convolutional models analysed in this work. For the hidden units, all the networks used a rectifier linear unit (ReLU) non-linear activation function. The baseline DNN model (*DNN*) comprised 6 hidden layers with 2048 nodes each. The baseline CNN model (*CNN*) followed Sainath et al. [10] and comprised two convolutional layers with kernel sizes 9×9 and 3×4 respectively. There were 256 feature maps in the first convolutional layer and 512 in the second one. In the baseline CNN model we used overlapping pooling and valid convolutions (no padding). In the first convolutional layer, the kernels were applied using a stride of 1. In the second convolutional layer, the kernel's strides were $s_{time} = 1$ and $s_{freq} = 4$.

Table 1. *Number of independent parameters (# params) and training times for Aurora 4 ($t_{Aurora4}$) and MGB-3 (t_{MGB-3}) datasets for the baseline models (DNN, CNN, VDCNN-max-4FC) and our VDCNN models (avg, max, max-addconv, all-conv).*

Model	# params	$t_{Aurora4}$	t_{MGB-3}
DNN	24.7M	8 h	57 h
CNN	23.8M	–	–
VDCNN-max-4FC	20.1M	17 h	155 h
VDCNN-avg	6.1M	55 h	147 h
VDCNN-max	6.1M	16 h	131 h
VDCNN-max-addconv	7.6M	31 h	–
VDCNN-allconv	7.6M	20 h	215 h

For all VDCNN models we use non-overlapping pooling, together with zero-padding at the output of each convolutional layer to ensure feature map size consistency within each CNN block. A CNN block is a stack of convolutional layers with kernel stride size $s_{time} = 1$ and $s_{freq} = 1$ (stride 1×1). In all our models we use 5 CNN blocks with 2-3 convolutional layers in each block. CNN blocks were alternated with down-sampling layers: max pooling, average pooling, convolutional layers reducing the feature maps size only in frequency dimension (stride 1×2), or convolutional layers reducing the feature maps size in both time and frequency (stride 2×2).

Our baseline VDCNN model *VDCNN-max-4FC* used max-pooling; the 2D output of the last pooling layer was transformed into a vector and fed into three FC hidden layers before the softmax output layer. The remaining VDCNN architectures that we investigated did not use any FC hidden layers after the final downsampling layer. In *VDCNN-avg* the max-pooling layers were replaced with average pooling using the same kernel sizes and strides as in the baseline VDCNN model; the output of the last pooling layer is fed directly into the softmax output layer. *VDCNN-max* differed from *VDCNN-avg* by using max-pooling in place of average pooling, whereas *VDCNN-allconv* used convolutional layers instead of pooling layers to perform downsampling. The strides of the kernels in the down-sampling convolutional layers matched the strides of the pooling layer kernels in our other VDCNN models, and the number of output channels of downsampling convolutional layers matched the number of their input channels. Introducing downsampling convolutional layers into the model resulted in an increase of the overall number of parameters. Therefore, to compare networks of similar complexity, the *VDCNN-max-addconv* model contains an additional convolutional layer in each convolutional block. The numbers of weights for each model are summarized in Table 1. By removing FC layers from the network architectures, our proposed VDCNN models greatly reduce the number of the independent parameters of the networks. In Table 1 we also show the training times for the models trained on the same GPU (Nvidia TITAN X).

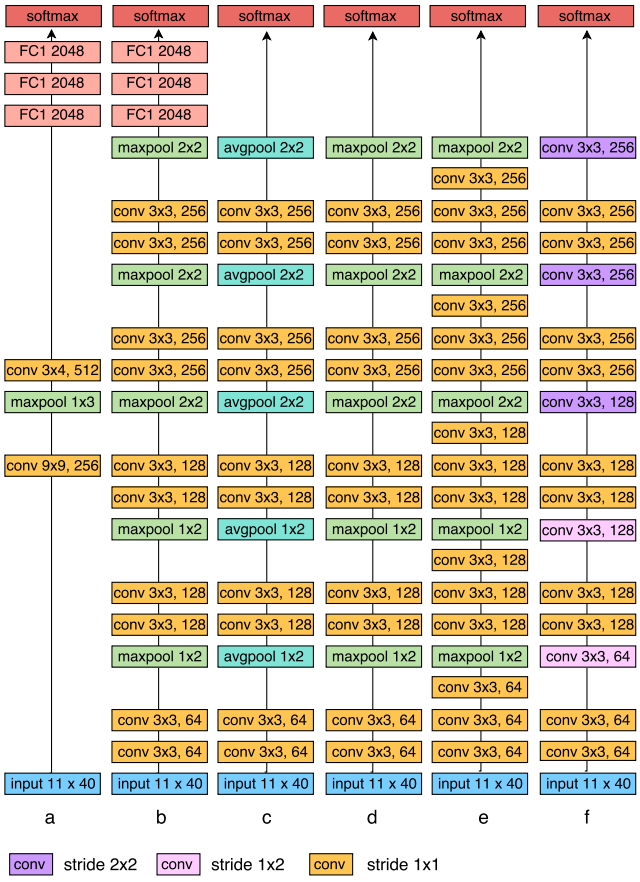

Fig. 1. Architectures of the convolutional models: a) *CNN*, b) *VDCNN-max-4FC*, c) *VDCNN-avg*, d) *VDCNN-max*, e) *VDCNN-max-addconv*, f) *VDCNN-allconv*.

All models were trained using the cross-entropy criterion, optimised with minibatch stochastic gradient descent using the Adam algorithm to smooth the gradient estimates. To keep the scale of the gradients and activations roughly the same in all layers we used "Xavier" initialization [26] for the weights. We seeded the random number generator to a nonnegative integer in all our experiments. The initial learning rates $\gamma \in [0.002, 0.001, 0.0008, 0.0005]$ were individually adapted for each model. The distribution of the outputs of each layer was normalized using batch normalization [27]. The training termination is conditioned on the development loss.

4. RESULTS

We evaluated the acoustic models on the Aurora 4 task and on the much larger MGB tasks. (The MGB-3 training set is over 20 times bigger than the Aurora 4 training set.)

Table 2. *Word Error Rates (WERs) [%] for Aurora 4 test sets A, B, C, D, and the average (AVG) WER for the baseline models (DNN, CNN, VDCNN-max-4FC) and our VDCNN models (avg, max, max-addconv, allconv) trained with alignments generated from multi-condition training set of Aurora 4.*

Model	A	B	C	D	AVG
DNN	3.47	7.67	7.85	19.73	12.55
CNN	3.33	6.89	6.59	17.92	11.34
VDCNN-max-4FC	2.43	5.92	5.74	16.26	10.09
VDCNN-avg	2.75	5.88	7.27	16.46	10.29
VDCNN-max	2.56	5.78	5.36	15.40	9.64
VDCNN-max-addconv	2.50	6.02	6.95	16.35	10.26
VDCNN-allconv	2.32	5.45	5.38	15.56	**9.55**

Table 3. *Word Error Rates (WERs) [%] for Aurora 4 test sets A, B, C, D, and the average (AVG) WER for different models trained with alignments generated from multi-condition training set (mc) and synchronized clean-condition training set (cln) of Aurora 4.*

Model	A	B	C	D	AVG
DNN/mc	3.47	7.67	7.85	19.73	12.55
DNN/cln	3.19	6.42	7.04	17.04	**10.79**
VDCNN-max-4FC/mc	2.43	5.92	5.74	16.26	10.09
VDCNN-max-4FC/cln	2.54	5.33	4.61	13.77	**8.70**
VDCNN-allconv/mc	2.32	5.45	5.38	15.56	9.55
VDCNN-allconv/cln	2.43	4.43	4.50	12.50	**7.75**

4.1. Evaluation on Aurora 4

We trained Aurora 4 models using two different alignments. The results in Table 2 were obtained with alignments produced by a GMM-HMM systems trained using the multi-condition Aurora 4 training set. We prefer this approach as it better matches the usual condition in which synchronized clean-condition training sets are not available for a task. However, in order to compare our results with other Aurora 4 results in the literature (which often use clean-condition training to generate the alignments for the NN-HMM systems), Table 3 compares some of our multi-condition alignment results with the results using clean-condition alignments for NN training.

Our baseline models – *DNN, CNN, VDCNN-max-4FC* – achieve lower WERs than the similar models in the literature. For instance, the best VDCNN model in [28] achieves a WER of 8.81% (with clean-condition alignments) compared to our 8.70% for the same network architecture. Similarly, our DNN baseline gives to our knowledge the lowest DNN WER on Aurora 4 (10.79% compared to 11.11% in [18] and [28]). These improvements in accuracy may result from differences in our training procedure – using the Adam optimizer (rather than stochastic gradient descent with the NewBOB learning rate schedule) and batch normalization. The state-of-the-art WERs reported in [18] are 7.99% with auxiliary feature joint training, and 7.09% with further LSTM-RNN joint decoding. Our best VDCNN model, without any additional input features nor additional neural network models, achieves a WER of 7.75%.

We consider the four simplified VDCNN architectures for robust speech recognition trained using the more realistic targets obtained from multi-condition alignments (Table 2). These results indicate that removing the fully-connected layers from the VDCNN model (*VDCNN-max*) improves the accuracy for test subsets B, C, and D. The gain is the most prominent for the most corrupted test subset D, which indicates that this kind of architecture may be beneficial also for other datasets corrupted with noise and with mismatched channels. Removing the majority of the independent parameters of the network resulted in WER reduction over the whole test set. However, the *VDCNN-avg* model slightly outperforms the baseline VDCNN only for the test subset B; the overall WER for this model is higher than the baseline WER. This model performs the worst for subsets A and C. This may be explained by the smoothing properties of the average pooling layer which for relatively clean test subsets may cause the performance degradation. Among all models trained with multi-condition alignments, the average WER over test subsets A, B, C, and D is the lowest for the *VDCNN-allconv* model, consisting solely of convolutional layers and a softmax layer, and without any fully-connected hidden layers. The relative improvement over the corresponding baseline VDCNN model is 5%, with a 16% relative improvement over the baseline CNN, and 24% over the baseline DNN. The largest relative WER reduction was over subset B (additive noise) compared to the VDCNN baseline, making the proposed *VDCNN-allconv* architecture a promising choice for other noisy speech recognition tasks. By aggressively reducing the number of parameters of a network by removing all FC layers and by introducing five additional convolutional layers to the VDCNN model, the capacity of the model is increased without considerably increasing the amount of the model's weights. This may enable the model to perform better generalizations in training and as a result perform better on disrupted test sets. The final proposed model, *VDCNN-max-addconv*, is worse than the VDCNN baseline for all test subsets, although this experiment indicates that the performance gain for *VDCNN-allconv* model is not solely due to model size expansion.

For the clean-condition alignments, the relative improvement for most accurate model is 11% over the corresponding VDCNN baseline, and 28% over the DNN baseline (Table 3). These results indicate that the proposed *VDCNN-allconv* architecture benefits the most from the clean alignments setup. For clean alignments, the relative performance gains over our VDCNN baseline for test subsets A, B, C, and D are 4%, 17%, 2%, and 9%, respectively. The test subset B (with ad-

ditive noise) and subset D (with additive noise and microphone mismatch) are again the ones benefiting the most from the proposed VDCNN architecture. For the same models architectures, the relative improvements for clean-aligned over multi-aligned models are 19% for *VDCNN-allconv*, and 14% for the baseline *VDCNN-max-4FC*. The choice of the alignments therefore has a major influence on the final WERs in the Aurora 4 task.

4.2. Evaluation on MGB

To ensure the reliability of our results, and to scale to a larger, more realistic task, we evaluated our models on the MGB Challenge data. We used the MGB-3 training set to train our acoustic models and used the language model training data used for the the MGB-1 Challenge [22]. To provide calibration for the acoustic and language models used in the MGB task, we evaluate the models trained on MGB-3 training data on both the MGB-3 development set and on the MGB-1 test set. It should be noted, however, that our MGB-1 results can not be directly compared to the other results in the literature for MGB-1 task as the training set used in this work is MGB-3 which contains about 3.5× less audio data than the MGB-1 set, drawn from a different set of multi-genre TV programmes.

WERs for the MGB-3 dev set and the MGB-1 test set are summarised in Table 4. In those tasks we did not aim for best possible performance; our objective was to compare the proposed VDCNN models with a VDCNN baseline and with each other on a task larger than Aurora 4. (For instance, further reductions in WER would be obtained through sequence training, speaker adaptation, improved test set segmentation, and re-alignment.) For both MGB test sets we obtained the lowest WERs using our proposed *VDCNN-allconv* model which is in line with the Aurora 4 findings above. It also confirms that this kind of architecture can be beneficial for other robust speech recognition tasks, with small but consistent gains over the VDCNN – 1.0% WER absolute improvement for MGB-3 and 1.2% for MGB-1. The WERs for average and max pooling are comparable in both MGB tasks. We also show the results of rescoring our best acoustic model with a 4-gram language model, giving an additional 2.2% and 2.5% absolute improvement for MGB-3 and MGB-1 tasks, respectively.

Table 4. *Word Error Rates [%] for MGB-3 dev set and MGB-1 test set for the baseline model (VDCNN-max-4FC) and our VDCNN models (avg, max, allconv). The last VDCNN-allconv-4G model is rescored with a 4-gram LM. All models trained on MGB-3 training data*

Model	**WER**	
	MGB3-dev	MGB1-test
VDCNN-max-4FC	53.2	42.4
VDCNN-avg	52.4	41.4
VDCNN-max	52.5	41.4
VDCNN-allconv	**52.2**	**41.2**
VDCNN-allconv-4G	**50.0**	**38.7**

5. CONCLUSIONS

In this paper we investigated very deep CNN architectures for robust speech recognition. We explored several simplified architectures to investigate which VDCNN components are necessary to obtain state-of-the-art results for robust speech recognition task. We investigated different downsampling strategies – our experiments demonstrate that max-pooling performs better than average pooling for smaller, Aurora 4 database, but pooling layers are not necessary at all to achieve state-of-the-art results for speech recognition with VDCNNs. Instead, using convolutional layers with increased stride can effectively enable the model to learn the necessary invariances. The performance gains resulting from the choice of the downsampling approach vary depending on the dataset and the alignments used. In addition, removing fully-connected layers from a VDCNN architecture, typically used in speech recognition, contributed most to the performance gains in our experiments, especially for noisy test data. Our model consisting solely of fifteen 2D convolutional layers with the same kernels sizes throughout the network and a single softmax classification layer gives the best performance consistently in our experiments on the Aurora 4 and MGB tasks.

In future work we want to aim for better MGB-3 task performance, including a more through hyperparameter search for this task. We plan to investigate the use of more information at the input of our simplified *VDCNN-allconv* network by expanding the width of the time context window, using delta and delta-delta FBANK feature maps as additional input channels, and increasing the number of frequency bins used in FBANK computation. Also, we shall explore training with auxiliary features (e.g. fMLLR) as the choice of input features for 2D convolutional layers is limited due to local correlation requirements.

6. ACKNOWLEDGEMENTS

This work was supported by a PhD studentship supported by The DataLab Innovation Centre, Ericsson Media Services, and Quorate Technology.

7. REFERENCES

[1] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. 2012.

[4] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3626–3633.

[5] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[6] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1533–1545, 2014.

[7] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E. Dahl, George Saon, Hagen Soltau, Tomás Beran, Aleksandr Y. Aravkin, and Bhuvana Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 2013, pp. 315–320, http://arxiv.org/abs/1309.1501.

[8] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014.

[9] Jui-Ting Huang, Jinyu Li, and Yifan Gong, "An analysis of convolutional neural networks for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4989–4993.

[10] Tara N Sainath, Brian Kingsbury, George Saon, Hagen Soltau, Abdel rahman Mohamed, George Dahl, and Bhuvana Ramabhadran, "Deep convolutional neural networks for large-scale speech tasks," *Neural Networks*, vol. 64, pp. 39 – 48, 2015, Special Issue on Deep Learning of Representations.

[11] Kevin J Lang, Alex H Waibel, and Geoffrey E Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural networks*, vol. 3, no. 1, pp. 23–43, 1990.

[12] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts.," in *Interspeech*, 2015, pp. 3214–3218.

[13] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015, http://arxiv.org/abs/1409.1556.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9, http://arxiv.org/abs/1409.4842.

[15] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun, "Very deep multilingual convolutional neural networks for LVCSR," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4955–4959.

[16] Tom Sercu and Vaibhava Goel, "Advances in very deep convolutional neural networks for lvcsr," in *Interspeech 2016*, 2016, pp. 3429–3433, http://arxiv.org/abs/1604.01792.

[17] Dong Yu, Wayne Xiong, Jasha Droppo, Andreas Stolcke, Guoli Ye, Jinyu Li, and Geoffrey Zweig, "Deep convolutional neural networks with layer-wise context expansion and attention," in *Proc. Interspeech*, 2016, pp. 17–21.

[18] Yanmin Qian and Philip C. Woodland, "Very deep convolutional neural networks for robust speech recognition," in *2016 IEEE Workshop on Spoken Language Technology, SLT 2016-Proceedings*, 2017, pp. 481–488, http://arxiv.org/abs/1610.00277.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778, http://arxiv.org/abs/1512.03385.

[20] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR (workshop track)*, http://arxiv.org/abs/1412.6806.

[21] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. 2011, IEEE Signal Processing Society.

[22] Peter Bell, Mark JF Gales, Thomas Hain, Jonathan Kilgour, Pierre Lanchantin, Xunying Liu, Andrew McParland, Steve Renals, Oscar Saz, Mirjam Wester, et al., "The MGB challenge: Evaluating multi-genre broadcast media recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 687–693.

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, http://tensorflow.org/.

[24] Vincent Renkens, "Kaldi with TensorFlow Neural Net," `https://github.com/vrenkens/tfkaldi`.

[25] Naveen Parihar and Joseph Picone, "Aurora working group: DSR front end LVCSR evaluation AU/384/02," *Tech. Rep.*, 2002, Institute for Signal and Information Processing, Mississippi State Univ.

[26] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[27] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456, http://arxiv.org/abs/1502.03167.

[28] Yanmin Qian, Mengxiao Bi, Tian Tan, and Kai Yu, "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2263–2276, 2016.